REMARKS

Claims 1, 2, 5, 7, 8, 13-15, 18-22, and 24-26 are currently pending. Claims 2, 5, 8, 11, 16 17, and 23 have been cancelled without prejudice. Claim 1 has been amended to incorporate the subject matter from canceled claims 2, 5, and 16. Claim 7 has been amended to incorporate the subject matter from canceled claims 8, 11, and 17. Claim 22 has been amended to incorporate subject matter from canceled claim 23 as well as subject matter now recited in claims 1 and 7. Claims 24-26 are newly added and are supported by, for example, the first and second paragraphs of the detailed description. It is respectfully submitted that no new matter has been added.

The Patent Office rejected claims 1, 2, 5, 7, 8, 11, and 13-23 under 35 U.S.C. 103(a) as being unpatentable over Shear, U.S. Patent No. 6,157,721, in view of Bodrov, U.S. Patent No. 6,802,006.

Applicant has disclosed the following in the background of the invention:

> In the field of this invention it is known that `Java 2` includes a significantly enhanced security model, compared to previous Java Virtual Machines (JVMs). This new model can restrict the behaviour of a Java applet or application to a clearly defined set of safe actions. This allows a user to download application code from the internet or across a computer network, and run the code in a previously installed JVM. The user can be confident that the application will be assigned the required privileges to function correctly, but to neither damage the user's machine nor divulge sensitive information held on that machine to others via the network or internet. **However, a problem with this approach is that the JVM itself must retain its security integrity in order to ensure downloaded code is restricted in this way. If a malicious user (hacker) has been able to gain access to the user's machine outside of the JVM environment and alter the behaviour of the JVM the whole Java security model is undermined.** For example, the hacker could alter the privileges assigned for software code from a specific source, thereby allowing subsequently downloaded code from this source to function beyond the limits otherwise set by the JVM, and such enhanced privileges could easily be configured to compromise the security integrity of the user's machine. Similarly, the hacker could disable the security code altogether, or worst still insert destructive routines into the core of the JVM which could be activated by an external trigger, such as specific time/date, or when other (possibly harmless) code is being executed. It is clear that with this malicious

> activity, early detection of such a compromise of the JVM core would be very useful, and could prevent more serious subsequent damage. If a malicious user decides to attack a machine, the JVM is an obvious target due to its significance in relation to web-based applications, servers and the like. Therefore the security integrity of the JVM is a highly significant factor in the security of the computer as a whole. **A need therefore exists for a software verification system, method and computer program element wherein the abovementioned disadvantages may be alleviated.**

The present claimed invention, as expressed by independent claims 1, 7 and 22, is directed to providing security for the installation of a virtual machine, such as a Java Virtual Machine.

Claim 1 recites

> A verification system, comprising: a primary library file, the primary library file having a digital signature, **wherein the primary library file is a virtual machine dynamic link library file**; a loader program that obtains a digital signature key and further loads the primary library file, wherein, if a public key cannot be obtained via a virtual machine provider, the digital signature key is a hidden public key internal to the loader program and, if a public key can be obtained via the virtual machine provider, the digital signature key is the public key obtained via the virtual machine provider; and a plurality of secondary files referenced by the primary library file, each of the plurality of secondary files having a digital signature; wherein the loader program verifies and selectively loads the primary library file by comparing the obtained digital signature key with the digital signature of the primary library file, the primary library file subsequently verifying and selectively loading the plurality of secondary files by calling the loader program to compare the obtained digital signature key with the digital signature of each of the plurality of secondary files, wherein the computer software installation is a virtual machine installation, at least one tertiary file referenced by at least one secondary file of the plurality of secondary files, **wherein after successful verification and selective loading of the at least one secondary file, the at least one secondary file manages the verification and selective loading of the at least one tertiary file**, at least one administrator-configurable file and the digital signature key comprising a number of keys including a private key provided by an administrator, wherein the loader program verifies the digital signature of the at least one administrator-configurable file using the private key, **wherein the verification system verifies the authenticity of each element of a virtual machine installation and provides security integrity for the**

virtual machine it installs.

Claim 7 recites

> A verification method, the method comprising the steps of: launching a loader program arranged to load files; if a public key is available from an internet site of a virtual machine provider, using the public key as a digital signature key; if a public key is not available from the internet site of the virtual machine provider, using a hidden key as the digital signature key; using the loader program to verify the authenticity of a digital signature incorporated in a primary library file by comparing said digital signature with the digital signature key, **wherein the primary library file is a virtual machine dynamic link library file**; selectively loading the primary library file in dependence upon the successful verification of its digital signature; using the primary library file and the loader program to verify the authenticity of digital signatures incorporated in each of a plurality of secondary files by comparing them with the digital signature key; and, selectively loading the plurality of secondary files in dependence upon the successful verification of their digital signatures, including at least one tertiary file referenced by at least one secondary file of the plurality of secondary files, the method comprising the further steps of: after successful verification and selective loading of the at least one secondary file, **using the at least one secondary file to manage the verification and selective loading of the at least one tertiary file**, at least one administrator-configurable file and the digital signature key comprising a number of keys including a private key provided by an administrator, wherein the loader program further verifies and selectively loads the digital signature of the at least one administrator-configurable file using the private key, **wherein the verification method verifies the authenticity of each element of a virtual machine installation and provides security integrity for the virtual machine it installs.**

Claim 22 recites

> A verification system comprising: a virtual machine primary library file, **the virtual machine primary library file** having a digital signature; a loader program that obtains a digital signature key and further loads the virtual machine primary library file; and a plurality of secondary files referenced by the virtual machine primary library file, each of the plurality of secondary files having a digital signature; wherein the loader program verifies and selectively loads the virtual machine primary library file by comparing the obtained digital signature key with the digital signature of the virtual machine primary library file, the virtual machine primary library file subsequently verifying and selectively loading the plurality of

> secondary files by calling the loader program to compare the obtained digital signature key with the digital signature of each of the plurality of secondary files, wherein the computer software installation is a virtual machine installation, wherein, if a public key cannot be obtained via a virtual machine provider over the internet, the digital signature key is a hidden public key internal to the loader program and, if a public key can be obtained via the virtual machine provider, the digital signature key is the public key obtained via the virtual machine provider over the internet; at least one tertiary file referenced by at least one secondary file of the plurality of secondary files, **wherein after successful verification and selective loading of the at least one secondary file, wherein the at least one secondary file manages the verification and selective loading of the at least one tertiary file**; at least one administrator-configurable file; and the digital signature key comprising a number of keys including a private key provided by an administrator, wherein the loader program verifies the digital signature of the at least one administrator-configurable file using the private key, **wherein the system verifies the authenticity of each element of a virtual machine installation and provides security integrity for the virtual machine it installs.**

Applicant has identified problems with the current art; e.g., a hacker can alter the behavior of the JVM outside the JVM environment so as to undermine the whole Java security model (page 1, line 25, through page 2, line 4) such as by disabling the security code or by inserting destructive routines into the core of the JVM (page 2, lines 13-17). The present invention provides a scheme for verification of the authenticity of a JVM using digital signatures and offers advantages. These advantages include 1) enhanced security of the JVM, 2) greater user confidence in the correct function of Java applications, and 3) improved detection of incorrect or damaged JVM installations (page 9, line 20, through page 10, line 2, of Applicant's specification).

Shear discloses techniques for certifying load modules such as executable computer programs or fragments by a protected or secure processing environment (column 1, lines 25-28). Shear discloses (column 9, lines 42-51):

> FIG. 2 shows how a verifying authority 100 can prevent the problems shown in FIG. 1. In this example, authorized provider 52 submits load modules 54 to verifying authority 100. Verifying authority 100 carefully analyzes the load modules 54 (see 102), testing them to make sure they do what they are supposed to do and do not compromise or harm system 50. If a load module 54 passes the tests verifying authority 100 subjects it to,

> a verifying authority may affix a digital "seal of approval" (see 104) to the load module.

Shear further discloses (column 5, lines 10-25):

> A web of trust may stand behind a verifying authority. For example, a verifying authority may be an independent organization that can be trusted by all electronic value chain participants not to collaborate with any particular participant to the disadvantage of other participants. A given load module or other executable may be independently certified by any number of authorized verifying authority participants. If a load module or other executable is signed, for example, by five different verifying authority participants, a user will have (potentially) a higher likelihood of finding one that they trust. General commercial users may insist on several different certifiers, and government users, large corporations, and international trading partners may each have their own unique "web of trust" requirements. This "web of trust" prevents value chain participants from conspiring to defraud other value chain participants.

Bodrov shows a dynamic connection between two executable images 100, 200 (Figure 2). Bodrov, in Figure 1, shows an executable image 100 that is a data object and that is dynamically connectable with other executable images. Quoting from Bodrov (column 3, lines 12-24):

> The executable image 100 is a data object that can define by itself or in conjunction with other executable images, one or more software applications. The software applications may include, for example: a word processor, a database, a digital rights management system, a personal finance utility, a graphics tool, an Internet browser, a computer game, a communications program, an authorization program, an electronic wallet, a multi-media renderer or a contract manager. Furthermore, the executable image 100 is dynamically connectable with other executable images. For example, in an embodiment of the invention that is developed for use with the Windows 95, the executable image is a dynamic link library (DLL).

Neither Shear nor Bodrov disclose or suggest a virtual machine or verifying each element of a virtual machine installation and providing integrity for the virtual machine that is installed. Shear discloses machine code and machine instructions. Bodrov discloses a virtual distribution environment and does mention in columns 2 and 3 security concerns with Java's download and execute capability, but does not disclose or suggest a verification method or verification system for installing a virtual machine, such as a Java Virtual Machine.

Furthermore, neither of Shear and Bodrov discloses or suggests the limitation "wherein

after successful verification and selective loading of one of the at least one secondary file, the at least one secondary file is arranged to manage the verification and selective loading of the at least one tertiary file."

Thus, it is respectfully submitted that claims 1, 2, 5, 7, 8, 13-15, 18-22, and 24-26 are allowable over Shear and Bodrov.

Claims 14 and 15 recite "the virtual machine provider is accessed through an internet site to provide the public key." Even if Shear could be considered to provide a certificate, Shear, in the abstract, Figure 1, column 2, lines 33-40; column 3, lines 10-15 and 21-35; and column 5, lines 3-5, or elsewhere, does not disclose or suggest the virtual machine provider provides a public key through an internet site. Bodrov does not appear to remedy this deficiency. Thus, claims 14 and 15 are allowable over the prior art for this additional reason.

The Patent Office is respectfully requested to reconsider and remove the rejections of the claims 1, 2, 5, 7, 8, 11, and 13-23 under 35 U.S.C. 103(a) based on Shear in view of Bodrov, and to allow all of the pending claims 1, 2, 5, 7, 8, 13-15, 18-22, and 24-26 as now presented for examination. An early notification of the allowability of claims 1, 2, 5, 7, 8, 13-15, 18-22, and 24-26 is earnestly solicited.

Serial No.: 10/050,083
Art Unit: 2137

Respectfully submitted:

Walter J. Malinowski October 31, 2007

Walter J. Malinowski Date

Reg. No.: 43,423

Customer No.: 29683

HARRINGTON & SMITH, PC
4 Research Drive
Shelton, CT 06484-6212

Telephone: (203) 925-9400, extension 19
Facsimile: (203) 944-0245
email: wmalinowski@hspatent.com

CERTIFICATE OF MAILING

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Commissioner for Patents, P.O. BOX 1450, Alexandria, VA 22313-1450.

_____ _____

Date Name of Person Making Deposit